Лабораторна робота 3

Дзьобан Максим

Мета

Імплементувати клас, що отримує вираз регулярного виразу (regex) на вхід та компілює для нього скінченний автомат (FSM). Далі скомпільований автомат можна використовувати для перевірки стрічок.

Підтримується обробка символів `*`, `+`, `.`.

Застосовується патерн проєктування *State design pattern* з використанням об'єктно-орієнтованого програмування для підтримки різних літералів.

```
regex_compiled = RegexFSM("a*4.+hi")
```

Клас `RegexFSM` у своєму конструкторі (`__init__`) приймає регулярний вираз у вигляді рядка, обробляє його символ за символом і на основі цього будує набір станів FSM. У тому числі враховуються символи `*`, `+`, `.`.

Створюються відповідні об'єкти станів:

- `AsciiState`,
- `DotState`,
- `StarState`,
- `PlusState`.

Кожен стан додається до списку `self.states`. FSM завершується `TerminationState`.

Метод `check_string`

Виконує наступні дії:

- Проходження по символах рядка та станах FSM.

- Обробку повторів, спецсимволів, буквальних символів згідно з FSM.
- Повертає `True` або `False` залежно від того, чи рядок відповідає регулярному виразу.

Основні класи

- `State` — абстрактний базовий клас для станів.
- `AsciiState` — відповідає одному символу.
- `DotState` — відповідає будь-якому символу.
- `StarState`, `PlusState` — реалізують оператори `*` та `+`.
- `StartState` та `TerminationState` — початковий і фінальний стани.
- `RegexFSM` — інтерпретатор регулярного виразу, що будує FSM та перевіряє рядки.

Висновок

- Реалізація відповідає вимогам: FSM будується на основі регулярного виразу, перевірка рядків працює.
- Абстракція станів застосована коректно.